Inverting the Formalization Workflow: Prototyping an MPC Protocol in Rocq with an LLM Agent

Cheng-Hui Weng
Nagoya University

Abstract

Formalization is usually the last step, but that ordering is costly when a construction spans unfamiliar mathematics. I invert the order and use formalization as a prototyping medium: an LLM drafts candidate models, Rocq checks them, and the human steers revisions. The case study is an ongoing formalization of SMC-PGG, a covering-space-based secure multi-party computation framework with 42 Theorem declarations. The main takeaway is methodological: prover feedback can guide cross-domain learning and design decisions early, before full pen-and-paper domain mastery.

Keywords. LLM-assisted formalization, Rocq, interactive theorem proving, secure multi-party computation, covering spaces.

Introduction. Formalization traditionally comes last. A researcher studies the relevant mathematics, designs a construction informally, and only then translates the result into a proof assistant as a final verification step. This ordering works well when the mathematical landscape is already familiar, but it becomes a serious obstacle when the construction draws on several domains that the researcher has not yet studied. In this work, I reverse the order and use formalization as a prototyping medium. An LLM agent[1] drafts formal models from natural-language direction, and a theorem prover checks whether each draft stands on solid logical ground. In this setting, the human learns from the structured output of each iteration. Definitions accepted by the theorem prover reveal useful abstractions, and failed proof obligations can signal missing assumptions, side conditions, or lemmas.

This case study contributes an inverted workflow ("idea → formalize → learn") and an ongoing Rocq formalization of SMC-PGG with 42 Theorem declarations, highlighting how human judgment, LLM drafting, and prover checks play distinct roles.

Inverted Workflow.

The starting point was a hypothesis from separate ongoing work on generalizing secure multi-party computation (SMC). The hypothesis was that any surjective relation can serve as a secret-sharing mechanism if legitimate parties possess a trapdoor for reconstruction. Covering spaces in algebraic topology fit this pattern. An index in the fiber over a basepoint can represent the secret, the monodromy action can represent the computation, and a trapdoor inverting the covering map can represent reconstruction. However, even formulating such connections informally already requires time to understand each contributing domain, and developing them to the level of a publishable pen-and-paper argument takes much longer. Rather than waiting to master group theory, algebraic geometry, and coding theory in advance, I used formalization in Rocq/MathComp to test whether this connection makes sense on solid logical ground.

From that point onward, formalization served not as a final verification step but as an exploratory instrument. Rather than waiting for complete prior mastery of every contributing domain, I used proof development to test whether the core hypothesis could be realized as a concrete protocol with checkable correctness and security guarantees. With this framing, I first asked the agent to assess the mathematical plausibility of the secret-hiding assumption and to compile working notes on implementations for sharing, computation, and reconstruction. Using those notes, I then asked the agent to draft formalization plans and evaluate their feasibility against available Rocq infrastructure, especially MathComp [6] and Infotheo [7]. In parallel, the search expanded into spectral graph theory, algebraic geometry, and coding theory while I learned unfamiliar material. This process also exposed infrastructure boundaries that required explicit axiomatization. For instance, the Riemann–Hurwitz formula

[1] All LLM-assisted work reported here used Claude (Anthropic), specifically Claude Opus 4.6.

is central to linking group size with reconstruction constraints, yet a full mechanization would itself be comparable to a separate research project. When a direction appeared unlikely to meet the target guarantees, I required the agent to produce structured tabular retrospectives that explicitly listed obstacles and alternatives, preventing vague narrative evasion and forcing detailed justification of the underlying reasoning. Formalization thus functioned as a methodological filter, separating components ready for mechanization from those that should remain explicit assumptions at this stage. The next section shows how this filter shaped the concrete SMC-PGG case-study formalization.

Case Study. To test the inverted workflow, I used it to build SMC-PGG (Secure Multi-party Computation via Parametric Geometry Group), starting from the core idea that a covering-space model can hide a secret as an index in the fiber over a base point and evolve it through monodromy, formalized as a representation $\rho : G \to S_N$. Through a human–AI hybrid formalization loop, this idea grew into a parametric mathematical framework before any venue- or domain-specific targeting; AI-assisted literature search then revealed that the framework aligns naturally with card-based SMC protocol research. Concretely, the dealer places T shares (total share count) on sheet positions $\mathbb{I}_N$, and each protocol step applies $\rho(g) \in S_N$, reshuffling positions while preserving reconstruction semantics. The formalization combines security and (K, T)-threshold constraints (K: number of shares needed to reconstruct the secret) in one parametric model and proves privacy/search-space bounds.

The framework is validated on four instances: uniform shuffling [10], biased shuffling [9], a classical S_5 construction, and an $S_5 \times S_5$ construction. The first three fit the simpler five-sheet setting; the $S_5 \times S_5$ case crosses the group size/geometry boundary and forces positive dropout tolerance $T - K > 0$. This provides evidence that the central trade-off is realized on both sides of the boundary. Except those four instances, I also show an impossibility result: in the regular abelian setting, symmetry alone cannot deliver non-trivial privacy/security gain. Methodologically, the section also illustrates learning through formalization: stronger arguments were introduced when simpler proof paths failed. The current development contains 42 Theorem declarations across group theory, combinatorics, algebraic geometry, coding theory, and information theory.

Observations. This case study suggests a methodological distinction from current lines of automated-reasoning work that prioritize theorem-level automation, such as whole-proof generation, informal-to-formal translation, or benchmark solving [1, 2, 5]. The workflow reported here instead semi-automates framework construction: it supports exploration across domains, iterative lemma drafting, and composition of many proved components into one coherent protocol architecture. In this setting, the main challenge is not only closing isolated goals, but also deciding which abstractions should exist and how security, reconstruction, and algebraic constraints should fit together.

The project also surfaced a practical architecture lesson. During repeated pivots, some formally verified components became obsolete, over-general, or temporarily disconnected from the main proof path. Examples include a Cartier-Foata trace-counting theorem, a right-angled Artin group word-reordering module, Massey's secret sharing from linear codes, and a weight bound for codes on hyperelliptic curves. These compile and are available for future instances, but do not load-bear in the current security or reconstruction proofs. This is less a failure of proving than a normal software-architecture effect under discovery-driven research: the framework requires continuous housekeeping to deprecate stale artifacts, refactor interfaces, and keep dependencies and theorem roles explicit, so the final development remains clear, auditable, and internally consistent.

Related Work. LLM-assisted proving work emphasizes proof generation, translation, retrieval, and training [1, 2, 3, 4, 5]. In contrast, this project uses the LLM mainly for domain-knowledge transfer and design-space exploration inside a prover-checked loop.

Conclusion. The case study supports a simple claim: AI-assisted formalization can guide early-stage protocol design, not only late-stage proof search. In this mode, formalization functions as a prototyping filter that both validates and teaches, making "idea → formalize → learn" a practical research workflow. From this perspective, the next step is twofold: to mature the current prototype into a publication-ready development, and to test the same workflow on new ideas in other domains.

References

[1] E. First, M. N. Rabe, T. Ringer, and Y. Brun. Baldur: Whole-proof generation and repair with large language models. In *ESEC/FSE 2023*, 2023.

[2] A. Jiang et al. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *ICLR 2023*, 2023.
[3] K. Yang et al. LeanDojo: Theorem proving with retrieval-augmented language models. In *NeurIPS 2023 (Datasets and Benchmarks)*, 2023.
[4] P. Song, K. Yang, and A. Anandkumar. Lean Copilot: LLMs as copilots for theorem proving in Lean. *arXiv:2404.12534*, 2024.
[5] Google DeepMind. AlphaProof and AlphaGeometry 2. Technical report, 2024.
[6] The Mathematical Components Team. Mathematical Components, 2024. `https://math-comp.github.io`
[7] R. Affeldt et al. A Coq Formalization of Information Theory and Linear Error-Correcting Codes, 2024. `https://github.com/affeldt-aist/infotheo`
[8] G. Barthe et al. Computer-aided security proofs for the working cryptographer. In *CRYPTO 2011*, LNCS 6841, Springer, 2011.
[9] D. H. Kim and A. Cetinkaya. Confidentiality in a Card-Based Protocol Under Repeated Biased Shuffles. *arXiv:2511.05111 [cs.CR]*, 2025.
[10] B. den Boer. More Efficient Match-Making and Satisfiability: The Five Card Trick. In *Advances in Cryptology — EUROCRYPT '89*, LNCS 434, pages 208–217, Springer, 1990.